Connecting the Dots between Parallel Programming and Energy

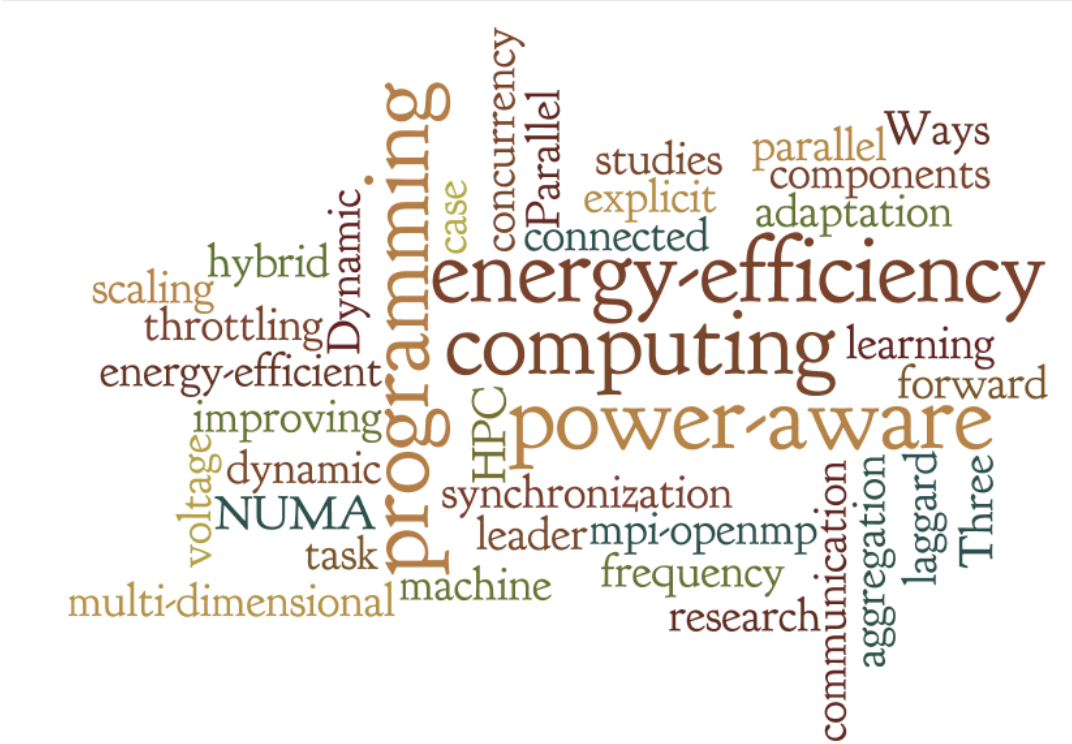
Connecting the dots between parallel programming and energy

Professor Dimitrios S. Nikolopoulos

d.nikolopoulos@qub.ac.uk

Points to get across

- PDP and HPC should be leading (not lagging) research on energy-efficient computing
- Parallel programming, runtimes systems and energy-efficiency are connected
- What lies ahead for the parallel programmer



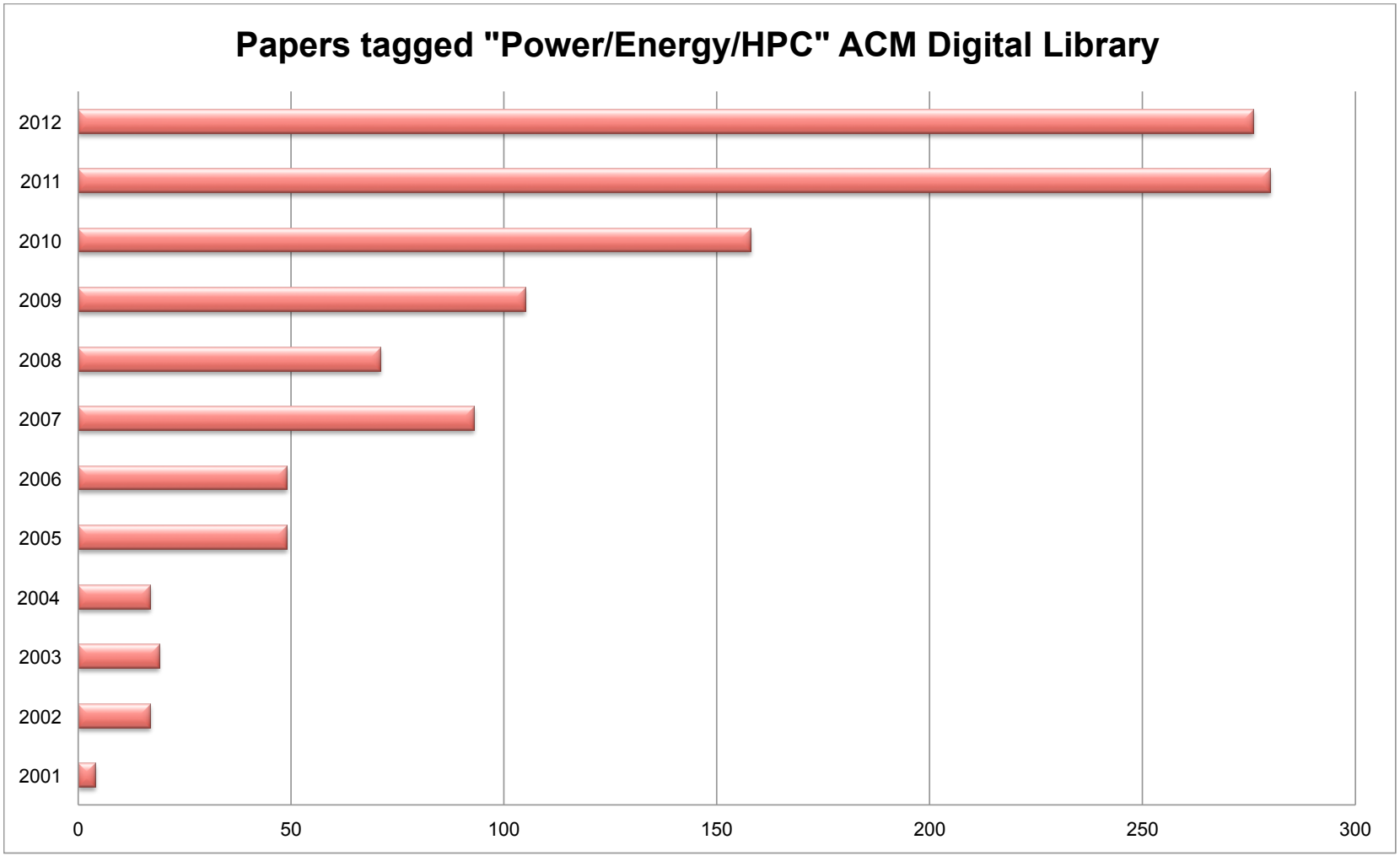
Acknowledgments

Our resources

EPSRC
Pioneering research and skills

IBM

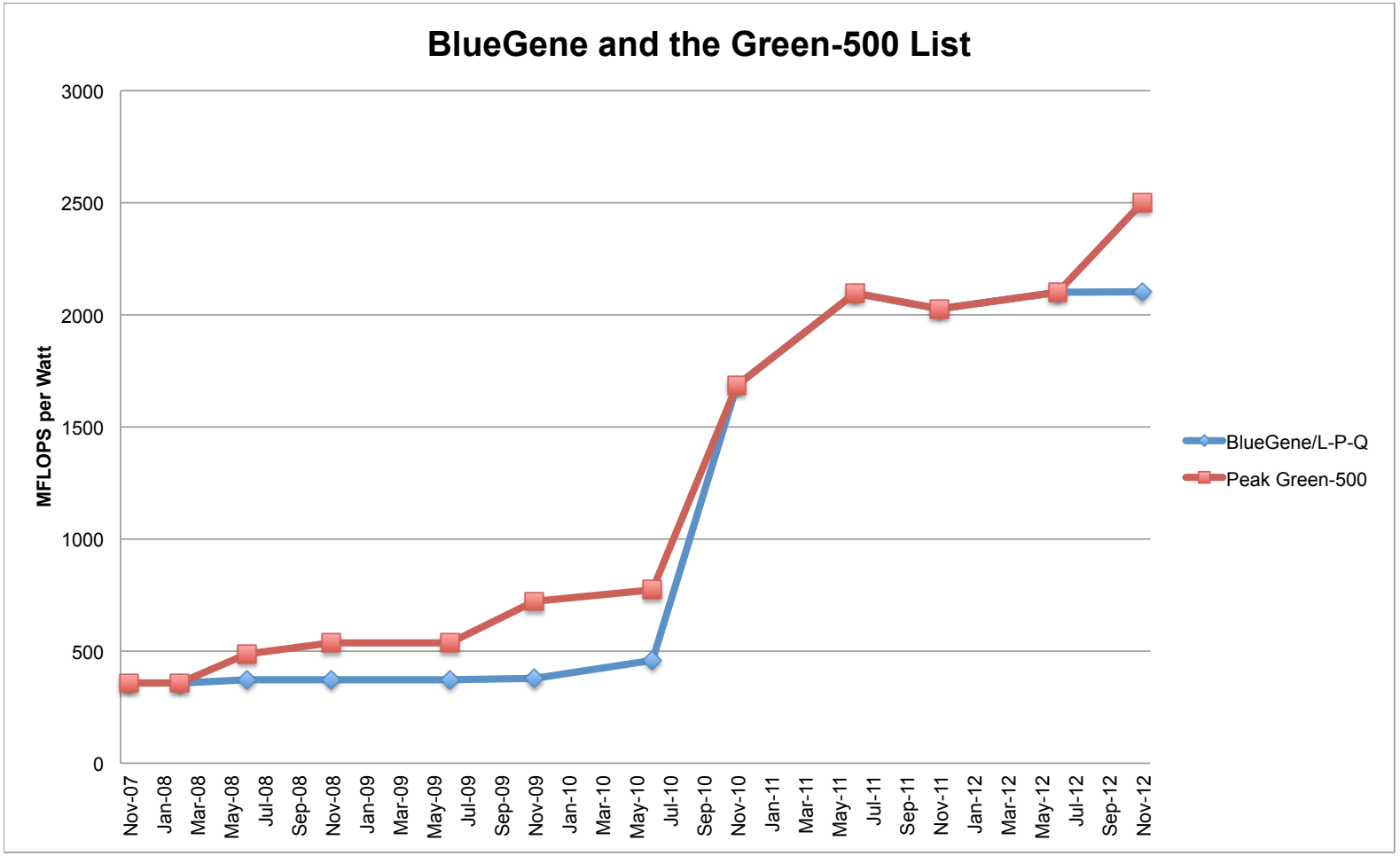
Energy in HPC



Leader or laggard?

- The history of BlueGene
  - Based on processors for the embedded systems market (PowerPC)
  - Pioneered "scale-out" idea, now common in datacentres
    - Many nodes with simple cores, custom interconnect
  - Dominated Top-500 list for performance

BlueGene on the Green500

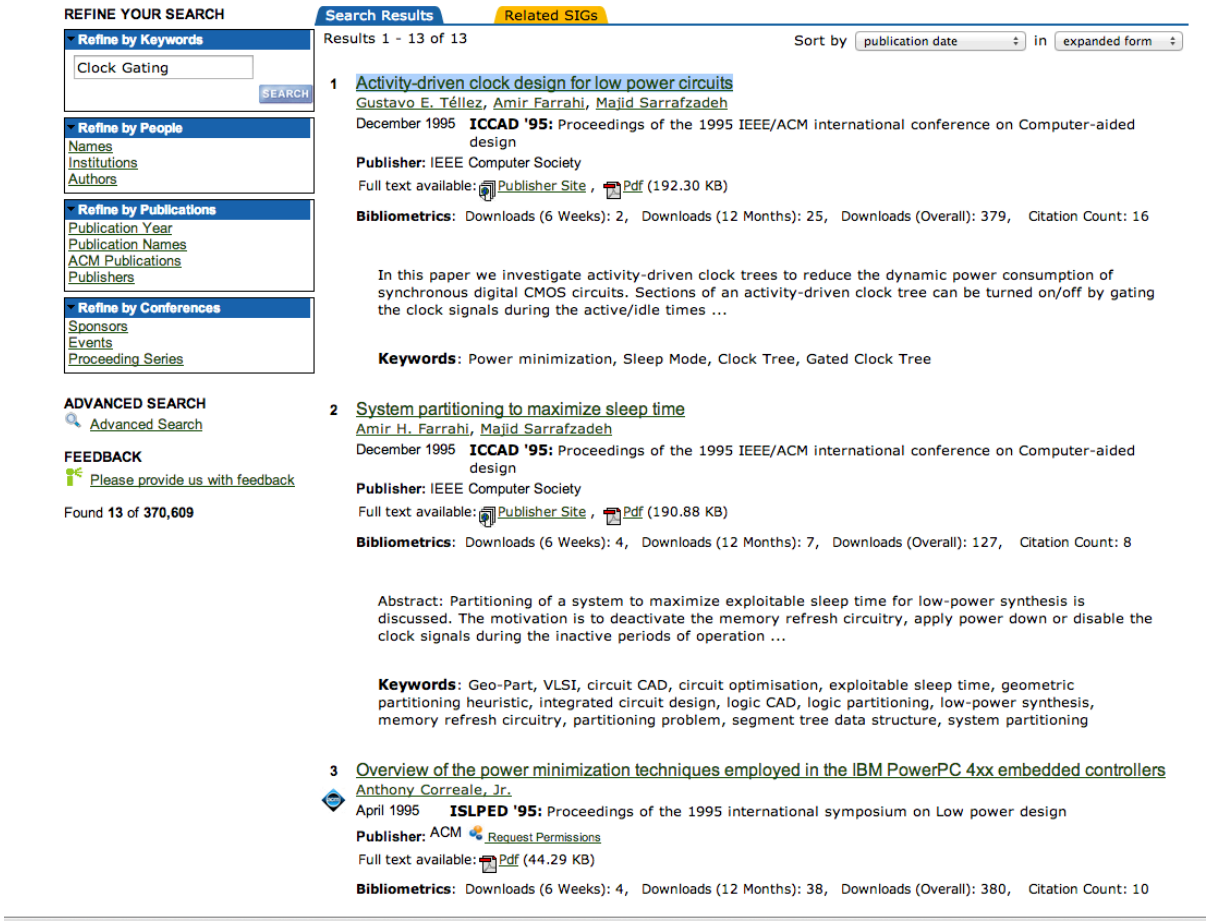


Leader or laggard?

Processor	Type	GFLOPS (32bit)	GFLOPS (64bit)	Watt (TDP)	GFLOPS/Watt (32bit)	FLOPS/Watt (64bit)
Adapteva Epiphany-IV	Epiphany	100	N/A	2	50	N/A
Movidius Myriad	ARM SoC: LEON3+SHAVE	15.28	N/A	0.32	48	N/A
ZiiLabs	ARM SoC	58	N/A	?	20?	N/A
Nvidia Tesla K10	X86 GPU	4577	190	225	20.34	?
ARM + MALI T604	ARM SoC	8 + 68	N/A	4?	19?	N/A
NVidia GTX 690	X86 GPU x 2	5621	234?	300	18.74	0.78
GeForce GTX 680	X86 GPU	3090	128	195	15.85	0.65
AMD Radeon HD 7970 GHz	X86 GPU	4300	1075	300+	14.3	3.58
Intel Knight's Corner (Xeon Phi)	X87?	2000?	1000	200?	10?	5?
AMD A10-5800K + HD 7660D	X86 SoC	121 + 614	?	100	7.35	?
Intel Core i7-3770 + HD4000	X86 SoC	225 + 294,4	112 + 73.6	77	6.74	2.41
NVIDIA CARMA (complete board)	ARM + GPU	? + 200	?	40	5.00	?
IBM Power A2	Power CPU	204?	204	55	3.72?	3.72
Intel Core i7-3770	X86 CPU	225	112	?	?	?
AMD A10-5800K	X86 CPU	121	60?	?	?	?

Leader or laggard?

- Is HPC just reusing solutions?
  - Processors originally designed for the mobile phones market
  - Clock gating, DVFS, device sleep states well known for 20 years



REFINE YOUR SEARCH

Refine by Keywords
Clock Gating
SEARCH

Refine by People
Names
Institutions
Authors

Refine by Publications
Publication Year
Publication Names
ACM Publications
Publishers

Refine by Conferences
Sponsors
Events
Proceeding Series

ADVANCED SEARCH
Advanced Search

FEEDBACK
Please provide us with feedback

Found **13** of **370,609**

Search Results | Related SIGs

Results 1 - 13 of 13 — Sort by publication date in expanded form

1 **Activity-driven clock design for low power circuits**
Gustavo E. Téllez, Amir Farrahi, Majid Sarrafzadeh
December 1995 **ICCAD '95:** Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design
Publisher: IEEE Computer Society
Full text available: Publisher Site , Pdf (192.30 KB)
Bibliometrics: Downloads (6 Weeks): 2, Downloads (12 Months): 25, Downloads (Overall): 379, Citation Count: 16

In this paper we investigate activity-driven clock trees to reduce the dynamic power consumption of synchronous digital CMOS circuits. Sections of an activity-driven clock tree can be turned on/off by gating the clock signals during the active/idle times ...

Keywords: Power minimization, Sleep Mode, Clock Tree, Gated Clock Tree

2 **System partitioning to maximize sleep time**
Amir H. Farrahi, Majid Sarrafzadeh
December 1995 **ICCAD '95:** Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design
Publisher: IEEE Computer Society
Full text available: Publisher Site , Pdf (190.88 KB)
Bibliometrics: Downloads (6 Weeks): 4, Downloads (12 Months): 7, Downloads (Overall): 127, Citation Count: 8

Abstract: Partitioning of a system to maximize exploitable sleep time for low-power synthesis is discussed. The motivation is to deactivate the memory refresh circuitry, apply power down or disable the clock signals during the inactive periods of operation ...

Keywords: Geo-Part, VLSI, circuit CAD, circuit optimisation, exploitable sleep time, geometric partitioning heuristic, integrated circuit design, logic CAD, logic partitioning, low-power synthesis, memory refresh circuitry, partitioning problem, segment tree data structure, system partitioning

3 **Overview of the power minimization techniques employed in the IBM PowerPC 4xx embedded controllers**
Anthony Correale, Jr.
April 1995 **ISLPED '95:** Proceedings of the 1995 international symposium on Low power design
Publisher: ACM Request Permissions
Full text available: Pdf (44.29 KB)
Bibliometrics: Downloads (6 Weeks): 4, Downloads (12 Months): 38, Downloads (Overall): 380, Citation Count: 10

Still looking for a proper metric…

- FLOPS/Watt
- FLOPS/Watt/square inch
- Energy × Delay
- Energy × Delay × Delay
- Energy × Time to solution
- £, $, €…

- Or should we just measure execution time and forget about the rest?

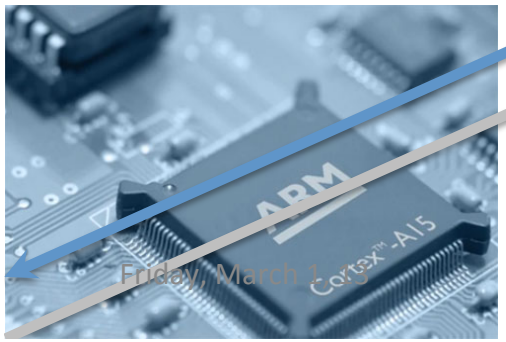
Is there a role for HPC & PDP in green computing innovation?

Exascale projections

- Assume that currently most energy-efficient supercomputer sustains improvement towards an Exaflop
  - Factor of 2384× in performance
  - 202.7 MegaWatt of power
- Assume target power cap of 20 MegaWatt
  - Need two orders of magnitude improvement in FLOPS/Watt
  - What hardware devices can achieve this improvement without compromising performance?

Parallel software is (or should be) energy-efficient

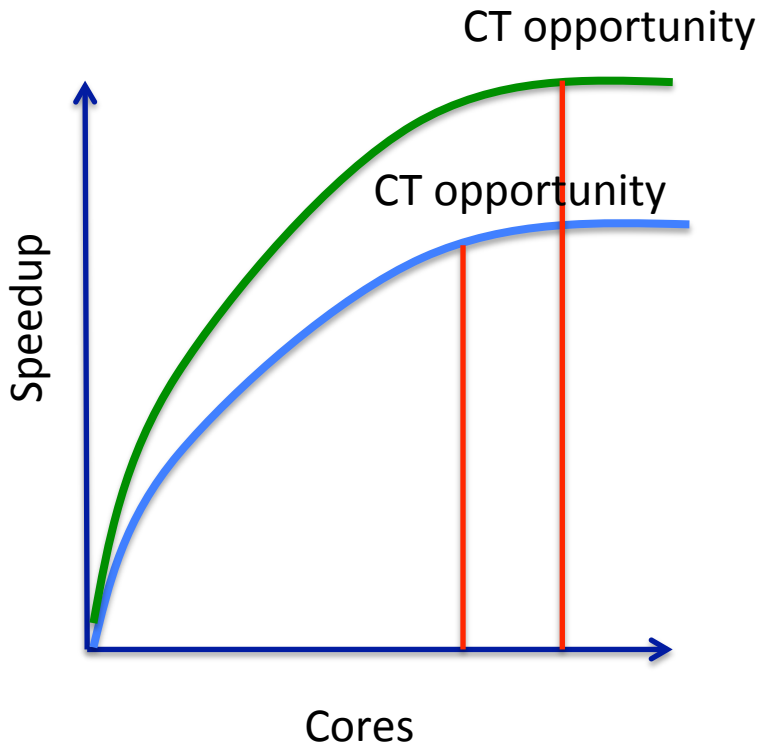
- Parallel software can scale up, down and out, if written with appropriate programming models
- Scale-free software can adapt to power constraints and control energy budgets



Dynamic concurrency throttling

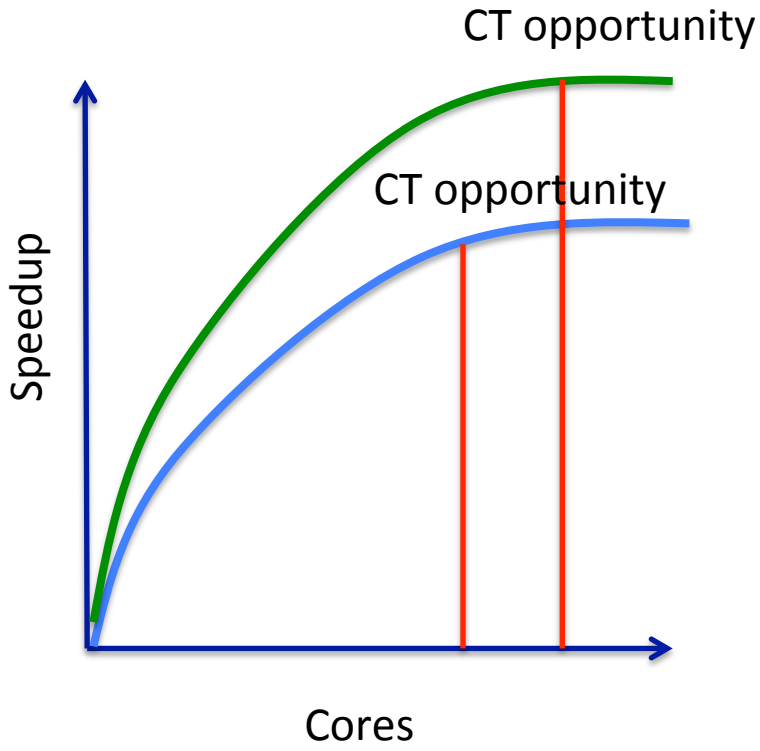
Concurrency throttling

- Dynamic power increases linearly with number of active cores
- Scalability curves have knees
  - Locate the knee
  - Energy-efficient parallel programs execute at the starting point of the knkee



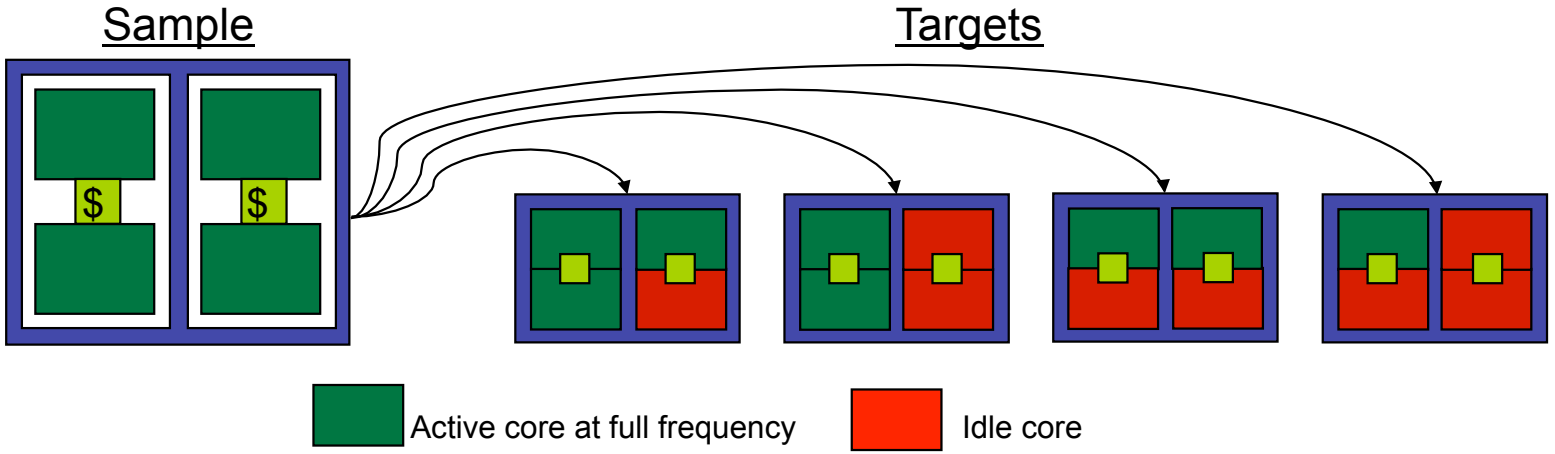
Exploiting the concurrency trade-off

- Programs execute distinct phases
  - Compute-, memory- or communication-bound
- Phase characterization indicates concurrency sweet spot
  - DCT can both improve performance and save power if the sweet spot is found
- DCT is one of several optimizations
  - Can easily complement DVFS to maximize energy-effiency



Unified program adaptation framework

- Multi-dimensional online performance predictor [Curtis-Maury et al., TPDS08, PACT08, ICS06]
  - Statistically analyzes samples of hardware event rates collected at runtime
  - Derives predictions based on data from short execution samples
- Runtime adaptation per program phase
  - Thread-based programming model (OpenMP, Cilk)
  - DCT and DVFS based on phase scaling predictions



Training an adaptation model

- Map event rates collected during sample configurations to predict performance on alternate, untested configurations

$$uIPC(t) = uIPC(s) \cdot a(e_1(s),\ldots,e_n(s)) + \beta$$

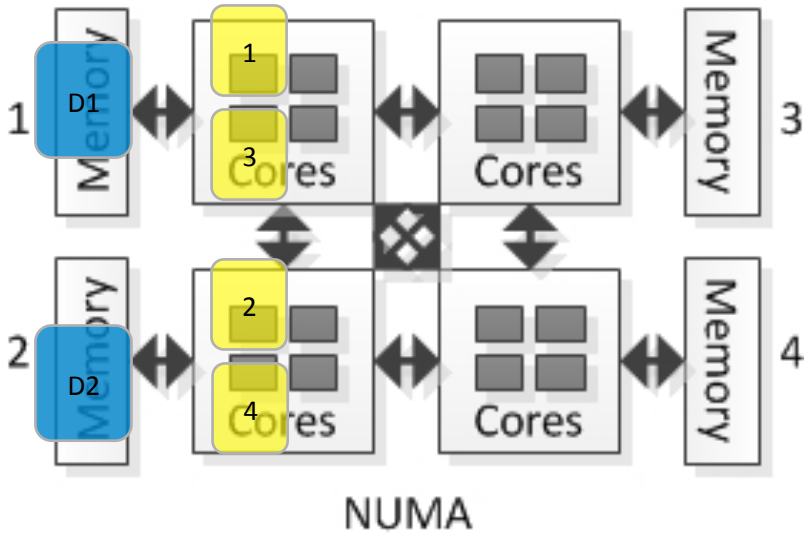
$$a = \sum_{i=1}^{n}(e_i(s) \cdot x_i + y_i) + z$$

$$uIPC(t) = uIPC(s) \cdot \sum_{i=1}^{n}(e_i(s) \cdot x_i) + uIPC(s) \cdot \lambda + \beta$$

- Regression, ANN, classifiers (e.g. SVM),…
- Simple models capture scalability curves well, albeit with lower absolute accuracy

Taming locality issues

- Original DCT failed to consider implications of cache refills, thread and data mapping



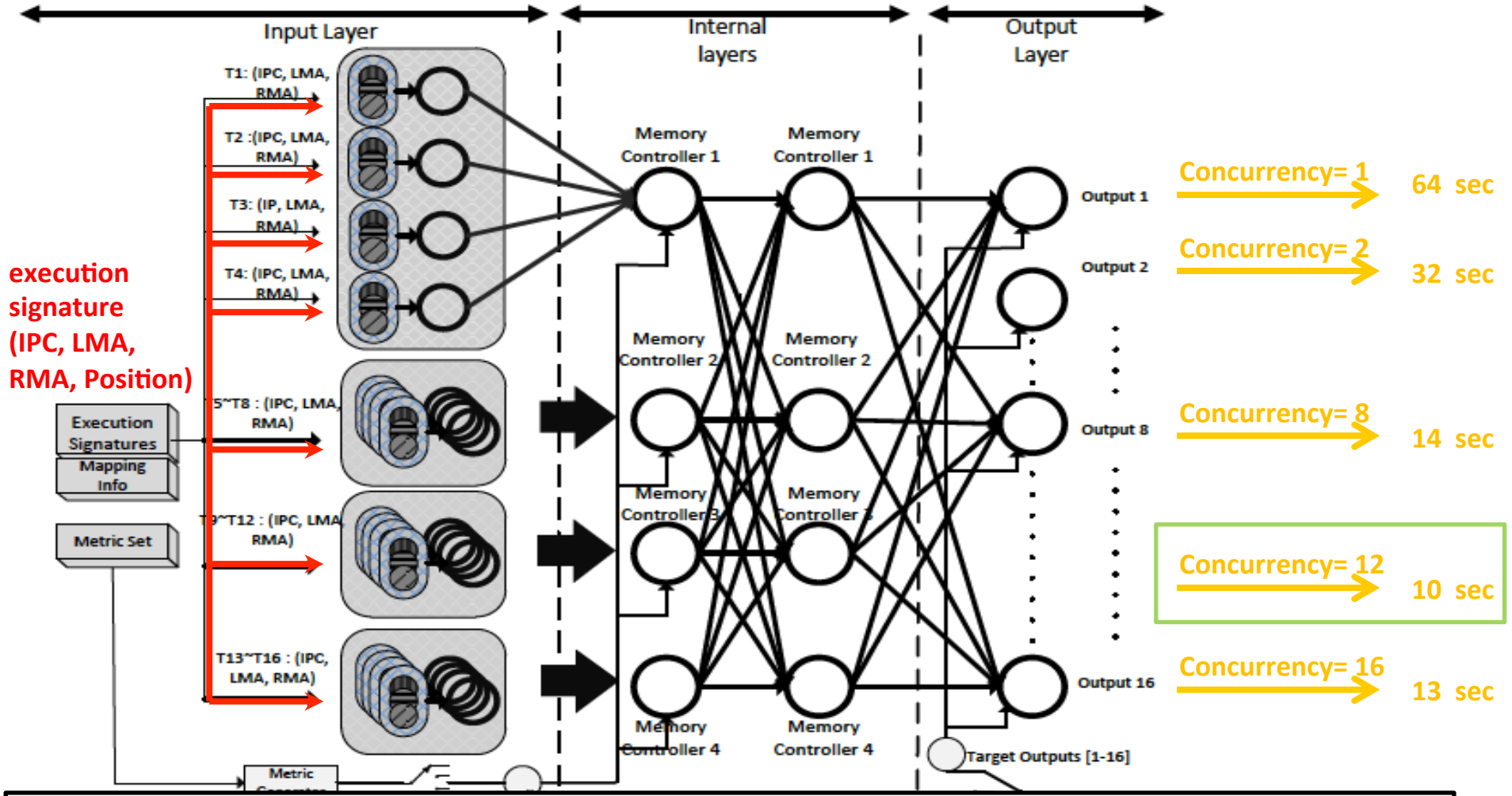
Example: 45% execution time variation across 85 mappings

4, 4-core nodes: 43,680 mappings.

16, 4-core nodes: 63 million mappings.

1000, 4-core nodes: 10^{43} mappings.

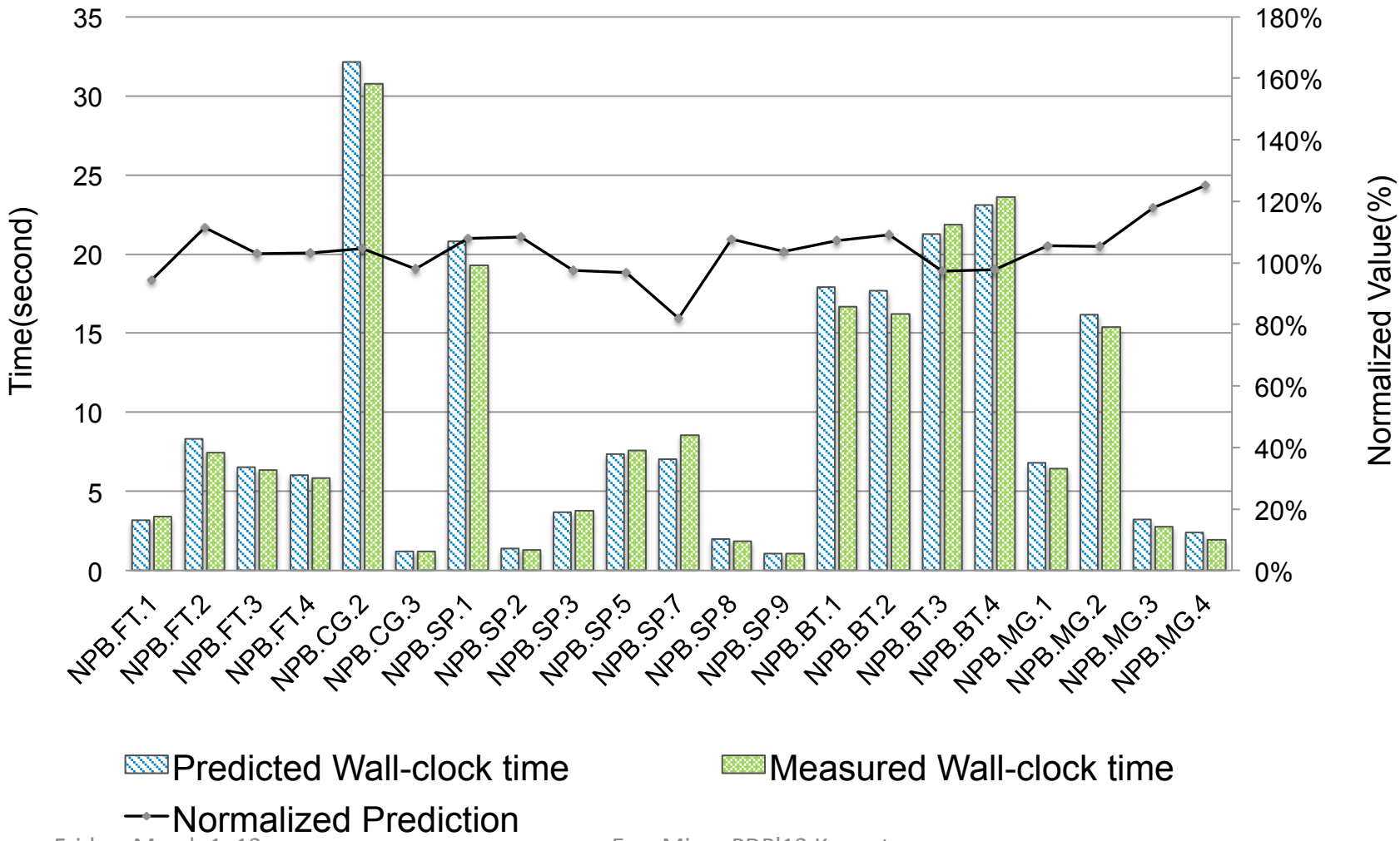
DyNUMA training using ANN



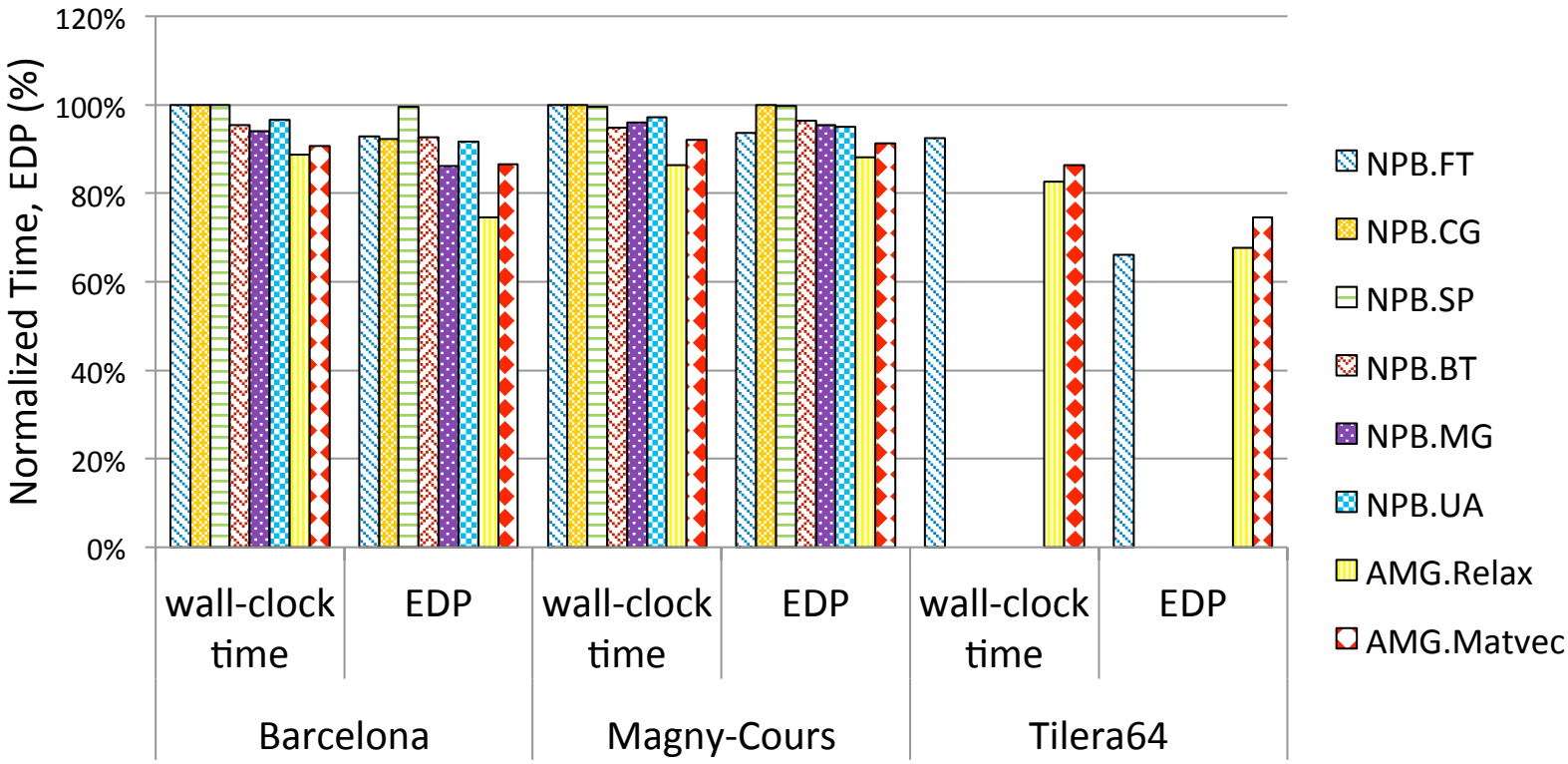
Optimize for concurrency, vertical and horizontal locality

[Su et al., IISWC12, SIGMETRICS PER12]

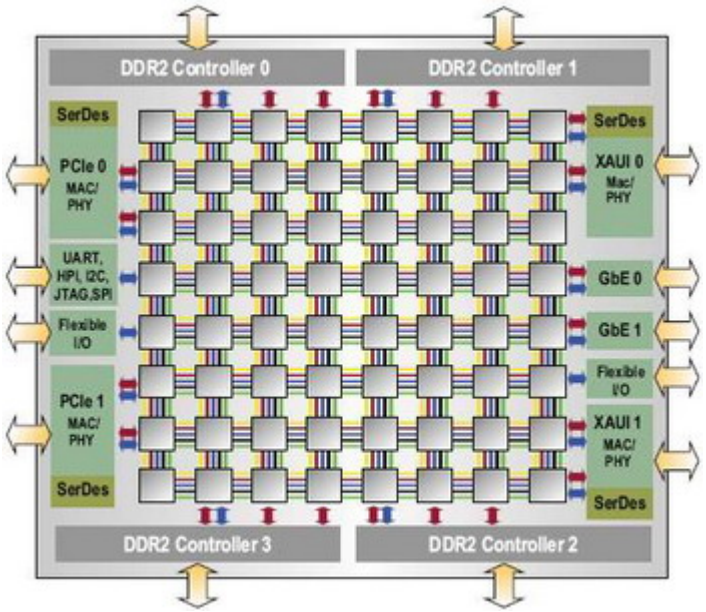
Modeling accuracy

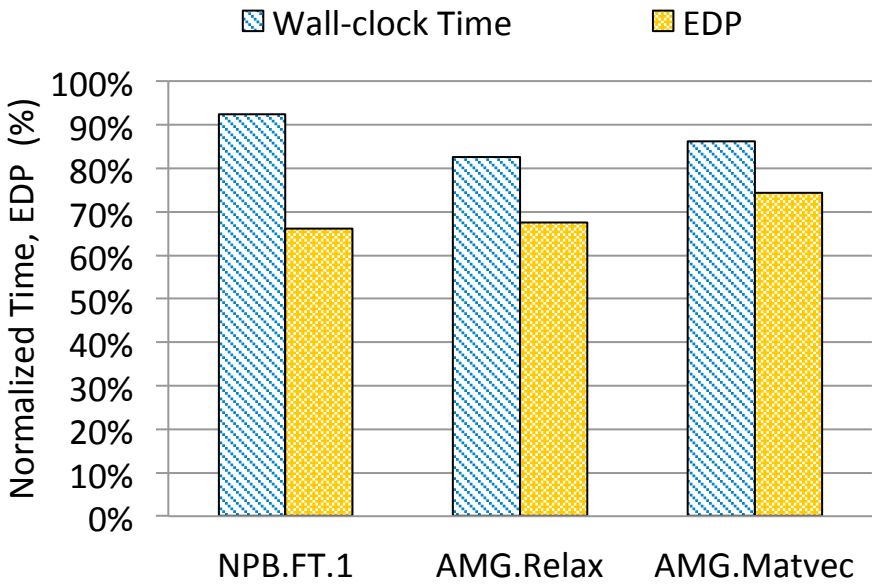


DyNUMA performance



Performance on TilePro64

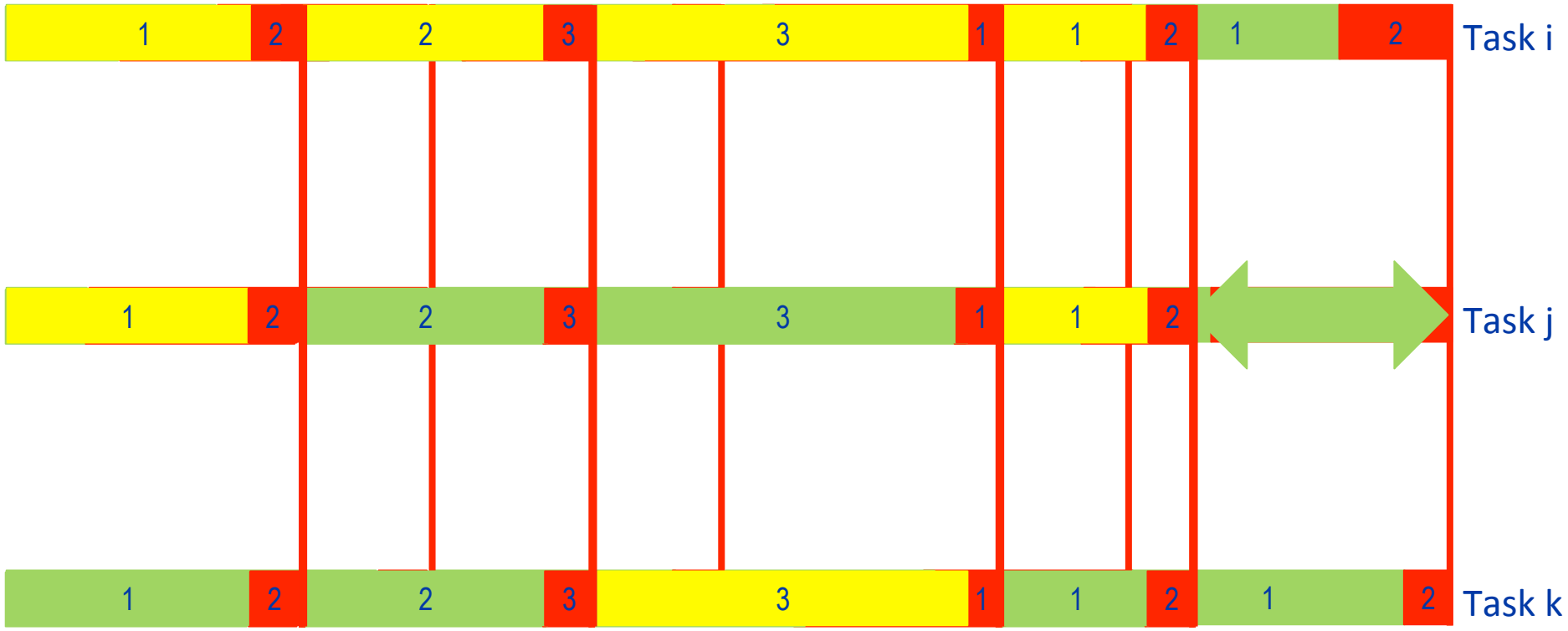




Tile64Pro OS default Linux mapping is inefficient
More concurrency does not necessary improve performance

Power-aware hybrid parallel programming

Energy-Aware Hybrid Programming



DVFS can save power and energy by reducing the CPU frequency and voltage

Critical path based modeling

- Predicting time vs. predicting scaling function

$$t_i = \sum_{j=1}^{M} \min_{1 \le |thr| \le X \cdot Y} t_{i,j,thr}$$

$$t_c = \max_{1 \le i \le N} \sum_{j=1}^{M} \min_{1 \le |thr| \le X \cdot Y} t_{i,j,thr}$$

Time modeling enables slack dispersion

- Slack dispersing DCT&DVFS (Li et. al, TPDS13)

Use critical path time to determine slack (essentially imbalance)

$$\Delta t_i^{slack} = t_c - t_i - t_i^{comm} - t_{dvfs}$$

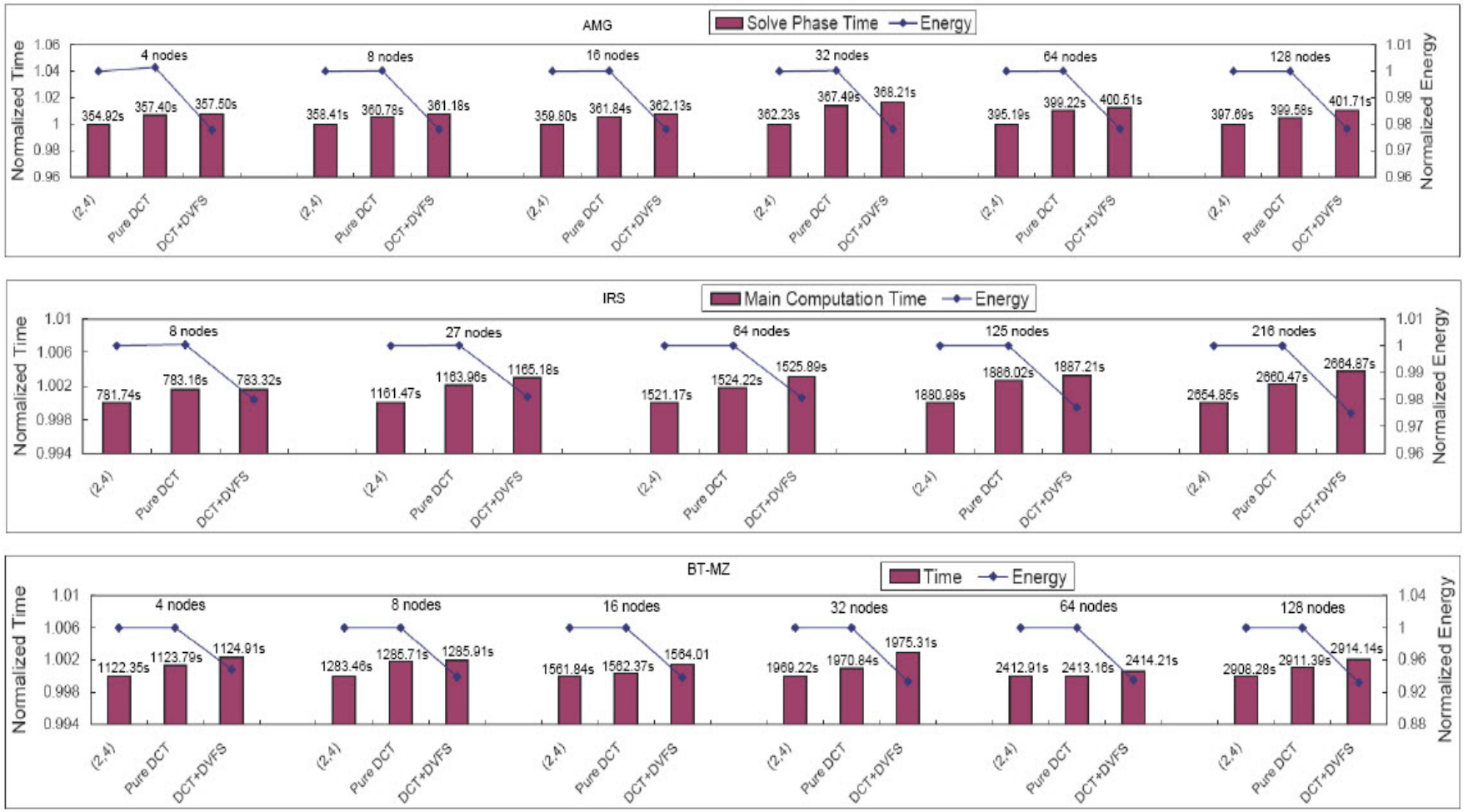
Time constraint:

$$\sum_{1 \le j \le M} \Delta t_{ijk} \le \Delta t_i^{slack}$$

Energy constraint:

$$\sum_{1 \le j \le M} t_{ijk} f_k \le t_i f_0$$

Performance Evaluation



Consistent (or improving) energy savings with weak scaling

Where is the connection?

Energy and the programmer

- Understand performance and power implications:
  - Thread mapping
  - Scheduling
  - Data placement
- Use adaptable, scale-free programming models
  - Phase-aware optimization infeasible otherwise
- Domain-specific knowledge can save energy
  - DAG, dataflow representations have enough information for a first-order approximation of the optimization problem
  - Proactive application-specific program control likely to outperform any system-level approach

Connecting the dots

- Parallel programs are unfortunately not prepared for a power crisis
- Programming models and runtime systems still are not (yet) up for the task
  - Adapt concurrency at runtime on non-coherent many-core architectures?
  - Control power states within the runtime with input from the application?
  - Scale-free, scale-aware?
  - How much waste is there in the runtime and the OS?

We educate two kinds of (parallel) programmers

Conclusive proof it pays more to do nothing than it does to get a Ph.D.:

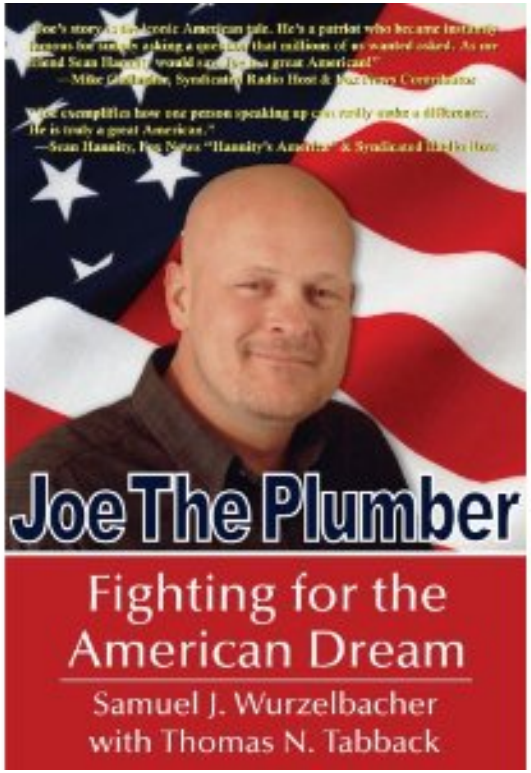
Average Maximum Annual Unemployment Benefit	Average Graduate Student Stipend
$21,060	$18,779

Sources: U.S. Department of Labor (via SF Chronicle), The Chronicle of Higher Education 2008-2009 survey of pay and benefits for teaching and research assistants. Unemployment benefits computed from average maximum state weekly benefits (typically 50% of base wages, capped by state) multiplied by 52 (in some cases, benefits can be extended up to 79 weeks). Academic year stipends extrapolated to 12 months.

JORGE CHAM © 2009

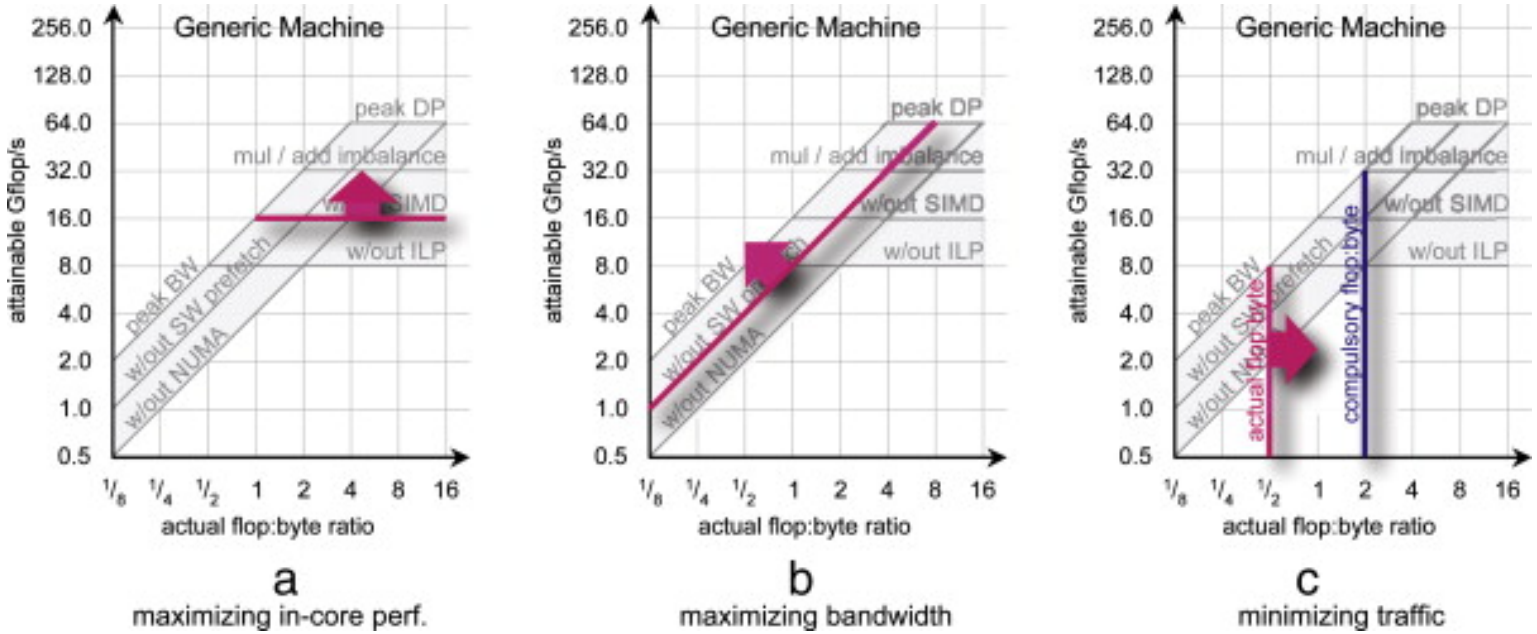
WWW.PHDCOMICS.COM

We educate two kinds of (parallel) programmers



Joe the ~~Plumber~~
Programmer

Revisiting parallel programming



- Environmentally conscious programmers know the hard limits of their algorithms given architectural or power constraints (e.g. DAG, roofline models)
- Environmentally conscious runtime systems eliminate waste
  - No more runtimes antagonizing operating systems
  - No more cross-runtime system noise

Revisiting parallel programming

- Your power budget may well be your most critical resource
  - Awareness of this issue must somehow become part of programmer education
  - Neither Joe nor graduate students are ready for this
  - Hardware and system software will not solve the problem by themselves, without a steep performance penalty
- Defensive parallel programming
  - Scale-free, react to power caps
  - Explicit resource proportioning

More information

http://www.qub.ac.uk/research-centres/HPDC/

Connecting the dots

- A power-aware parallel program of the future:
  - adapts concurrency
  - eliminates waste (idle time, redundant communication)
  - controls component power at a fine granularity (e.g. per task)

Manousakis et al, [SBAC-PAD12]